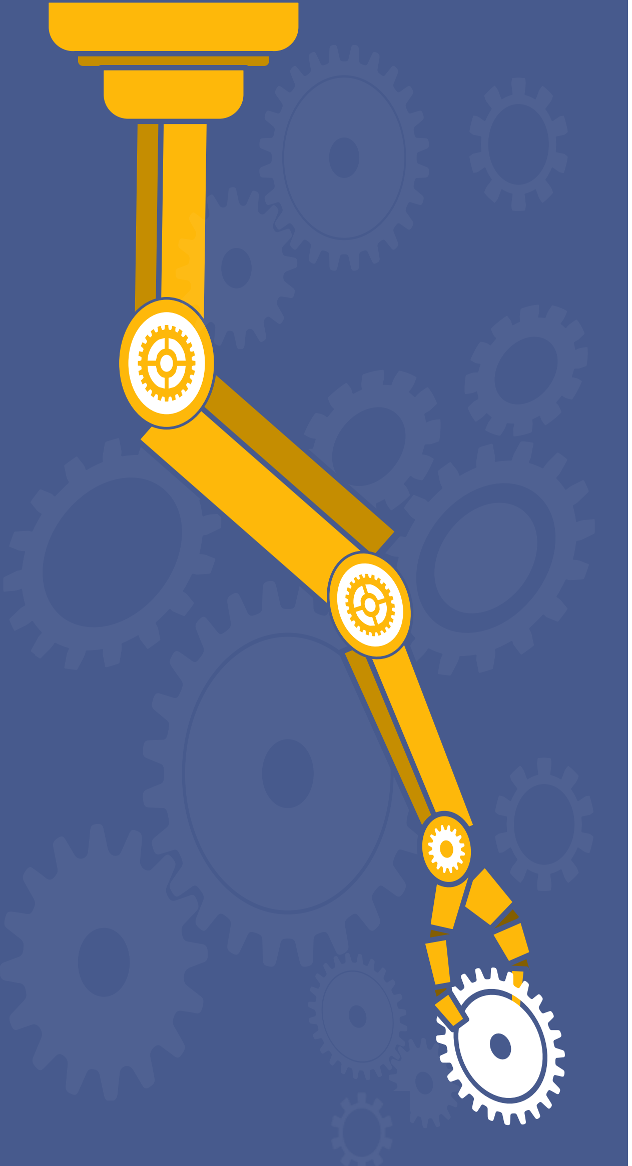


VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CSE2012- Design and Analysis of Algorithms

Dr. Iyappan Perumal

Assistant Professor Senior Grade 2

School of Computer Science & Engineering

VIT, Vellore.

Parenthesizing Arithmetic Expression

- Problem : Assume you have an unparenthesized arithmetic expression with only + and * operators. You can change the value of expression by parenthesizing at different positions. To keep it simple, assume that parenthesis occur only before or immediately after operands and not operators. Design an algorithm that can take a maximum possible value the expression can take in after adding the parenthesis.
- **Example Input : expr = “1+2*3+4*5”**
- **Minimum Value = 27**
- **Maximum Value = 105**
- **Minimum evaluated value=1+(2*3)+(4*5)= 27**
- **Maximum evaluated value=(1+2)*(3+ 4)*5 = 105**

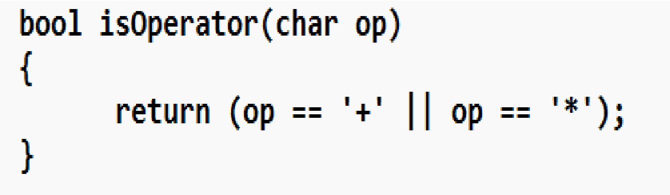
```
MinAndMaxValueOfExp(string exp)  //1+2*3
{
          vector<int> num; // stores the numbers from the given expression
          vector<char> opr;// stores the operators from the expression
          string tmp = "";
          // store operator and numbers in different vectors
          for (int i = 0; i < exp.length(); i++)
          {
                    if (isOperator(exp[i]))
                    {
                              opr.push_back(exp[i]);
                              num.push_back(atoi (tmp.c_str()));
                              tmp = "";

                    }
                    else
                    {
                              tmp += exp[i];

                    }
          }
          // storing last number in vector
          num.push_back(atoi(tmp.c_str()));

          len = num.size();
          minVal[len][len];
          maxVal[len][len];
```

Function to verify whether a character is operator symbol or not

used to push elements into a vector from the back

built-in function in which returns a pointer to an array that contains a null-terminated sequence of characters representing the current value of the basic string object

converts a character string to an integer value



```
bool isOperator(char op)
{
      return (op == '+' || op == '*');
}
```

// Initializing minval and maxval array (2D Array)

```
for (int i = 0; i < len; i++)
{
        for (int j = 0; j < len; j++)
                {
                        minVal[i][j] = INT_MAX;
                        maxVal[i][j] = 0;
                        // initializing main diagonal by num values
                        if (i == j)
                        minVal[i][j] = maxVal[i][j] = num[i];
                }
}
```

INT_MAX - an integer variable cannot store any value beyond this limit.

```
// looping similar to matrix chain multiplication and updating both 2D arrays
for (int L = 2; L <= len; L++)
{
    for (int i = 0; i < len - L + 1; i++)
    {
        int j = i + L - 1;
        for (int k = i; k < j; k++)
        {
            int minTmp = 0, maxTmp = 0;
// if current operator is '+', updating tmp variable by addition
            if(opr[k] == '+')
                minTmp = minVal[i][k] + minVal[k + 1][j];
                maxTmp = maxVal[i][k] + maxVal[k + 1][j];
// if current operator is'*',updating tmp variable by multiplication
            else if(opr[k] == '*')
                minTmp = minVal[i][k] * minVal[k + 1][j];
                maxTmp = maxVal[i][k] * maxVal[k + 1][j];
```

```
// updating array values by tmp variables
                    if (minTmp < minVal[i][j])
                              minVal[i][j] = minTmp;
                    if (maxTmp > maxVal[i][j])
                              maxVal[i][j] = maxTmp;
                    }  // End of k
          }  // End of i
} End of L

// last element of first row will store the result
          return minVal[0][len - 1];
          return maxVal[0][len - 1];


}
```